ECS 152A: Computer Networks Project 1B

Analyzing Pcap Files

Professor Zubair Shafiq

Krystal Chau, SID: 920918540 Jacob Feenstra, SID: 921423591

November 3, 2023

1 File Submissions

The following files were submitted along with this `proj1b` PDF.

1. `proj1b-1.py`
2. `proj1b-23.py`
3. `proj1b-1-ChatGPT.py`
4. `proj1b-23-ChatGPT.py`

2 Pcap Activities

1. Pcap 1

   A few packets in the first provided pcap have secrets in their contents. The secrets are obvious, and can be distinguished by their naming conventions (namely `secrets`, `secrets-2`, and so on). The prompt calls for only packets `to` the server, so we focus on request packets. We also noted that the secrets are contained in just HTTP, so we can search by these particular packets alone. The main problem is that many of these secrets are in different HTTP headers and elements: one is in the request line, another is in a header line, and a final one is in the JSON object encoded in the entity body. So between these three packets, we need something which can print out the packets of multiple format-types. Luckily, the `dpkt` library has a method in `http.py` known as `repr()`. As far as we can interpret it, it prints out the method line, header lines, and entity body of the HTTP request. In so doing we meet the activity requirement: to output each secret in a separate line.

2. Pcap 2 and 3

   The Pcap 2 and 3 exercise asks that we distinguish the activities of relevant packets, and note the differences in said activities between said pcaps. "Activities" is used loosely here, and the programming project does not convey the semantics of an "activity." We assume that this means the state/effect of the ICMP protocol. Since all of the packets follow `ICMP`, we can focus on the properties of this protocol. We have defined "activities" as the results of the ICMP response or request. To determine the results, we consulted RFC 792, which is the IETF-accepted definition of the ICMP protocol.

   In our analysis of the pcaps, we noticed that two differing elements of ICMP packets were a `type` and `code` identifier. We decided to look into the syntax of an ICMP, leading us to RFC 792, which explains in plain detail what these two identifiers mean.

   (a) `type` indicates the type of ICMP message. Whether or not it is a request, response, or something else.

   (b) `code` specifies the particularities of the given ICMP message. A type of message (`type`) has many slight variations, each of which is defined by a corresponding returning `code` in the packet.

So in our `proj1b-23.py` implementation, we analyze each ICMP packet's type and code values, and print them out. Then, referencing RFC 792, we can infer what each packet is doing (what the "activity" is) by it's type and code.

2.1 Pcap 2

(a) Packets 1-29: A `type` of 11 specifies a "time exceeded" ICMP message. This aligns with the Info header in Wireshark for packets in `ass1_2.pcap`: "Time-to-live exceeded Time to live exceeded in transit". The packet's time to live is exhausted before it reaches it's destination, and it is discarded by the router it is at. Furthermore, the packets have a `code` of 0, further cementing that the TTL was 0 when it arrived at a router, instead of some other subtlety. The TTL is measured in seconds and decrements as it transmits through a network end-to-end (according to RFC 792). Our guesswork analysis for the activity is that the destination is a legit end-system for the packet to reach, but it is not able to due to the TTL being 0. Looking at the TTL value for all of these packets, most have a value of 1, and a select few have a value of 0. (our python script prints it out). RFC 792 leads us to believe that this is 1 or 0 seconds, which according to the type and code we are receiving, is not enough time for the packets to arrive to the end-system.

(b) Packets 30-32: These last 3 packets have a different `type` and `code` than the 29 that has been analyzed so far. A `type` of 3 specifies a "destination unreachable" ICMP message. According to RFC 792, the router which generates this message has some way of determining the distance to the destination end-system. There are 6 codes, and so a few ways that the user might be thrown a "destination unreachable" message. These particular ICMP packets have `code` 3, which according to RFC 792, means that there is no active process port on the destination's system. From what we can gather, this means that these last packets `are` able to arrive at the destination address, but cannot be processed because the process port (assigned by the operating system) is not configured for ICMP. We believe these last few packets are rejected by the socket API, but can't base our findings. We surmise this is a different "activity" than what we see in the other packets.

2.2 Pcap 3

Pcap 3's analysis is identical to Pcap 2's. The number of packets in the capture differ slightly, but it otherwise has the same ICMP types and codes. 27 TTL exceeded packets, followed by 3 destination unreachable packets. Yet as the prompt indicates, the activities for these two captures differ slightly, despite all packets being the same 2 ICMP's. We believe it has to do with source-type between the two captures, and subsequently, the timeframe for each capture.

2.3 Pcap 2 versus Pcap 3

The main distinction we can note between the two captures is the MAC address. That is, the user system. For `ass1_2.pcap`, the source MAC address is `72:62:3f:a6:b3:de`. We assume this is a local, private machine used in the exercise. Whereas in `ass1_3.pcap`, the source MAC address is `CiscoSPV_91:f87f`. We believe that this is some sort of networking device on the edge of the network that is not a local machine; a user's computer. In each of these addresses, there is an `LG` bit listed. The MAC address for pcap 2 defines it as a "locally administered address (this is NOT the factory default)". The MAC address for pcap 3 defines it as a "globally unique address (factory default)". This further cements our intuition that the source machine for pcap 2 is a local machine, and the one for pcap 3 is some network device developed for industry use. What it particularly is, we are not sure.

The other notable difference is the timeframe for each capture. The globally unique address (pcap 3) is leaps and bounds faster than the locally administered address (pcap 2). It sends approximately the same number of ICMP packets in less than $\frac{1}{21}$ of the time! We think this is useful for the activity analysis: a source machine with better traffic handling, that can send packets at an expedient rate, generates the `same` ICMP message results as a much slower machine.

It is clear that there is something awry with the end-system that is being transmitted to. The end-system is the same for both pcap 2 and pcap 3: `Apple_7e:3a:ab`. We presume this is some server owned and operated by Apple. So the same exact server, regardless of two widely different source performances, results in the same ICMP messages.

To summarize, the difference between the two pcap's is the source medium employed: a slower, local machine versus a fast global machine. It is evident that they result in the same ICMP messages, so there is something happening on the destination-end. This our assumption for the "subtle difference." The difference is shown in `proj1b-23.py` by the timestamps and source address: two different sources, two different performances, and the same ICMP messages.

3 Statistics

1. List the unique source and destination IP addresses do you see in each pcap file?

ass1_1.pcap:	Source IP	2601:200:c100:f730:c07c:ff8a:3166:c648
	Destination IP	2606:2800:220:1:248:1893:25c8:19469

ass1_2.pcap:	Source IP -	192.168.80.223
		10.6.102.254
		10.5.102.254
		10.176.253.172
		10.166.72.77
		10.166.72.82
		10.177.13.84
		10.166.112.1
		152.195.84.214
		152.195.84.139
		152.195.84.141
		93.184.216.34
	Destination IP -	192.168.80.84

ass1_3.pcap:	Source IP -	10.0.0.1
		96.120.14.125
		96.110.222.141
		68.85.120.246
		96.110.45.237
		96.110.45.225
		96.110.33.10
		96.110.33.2
		50.242.151.138
		152.195.77.129
		93.184.216.34
	Destination IP -	10.0.0.252

2. For both pcaps, iterate over the packets and print the packet number, source ip address and destination ip address for each packet. The list you print should be sorted in ascending order of packet number.

	4, 2601:200:c100:f730:c07c:ff8a:3166:c648, 2606:2800:220:1:248:1893:25c8:1946
ass1_1.pcap:	18, 2601:200:c100:f730:c07c:ff8a:3166:c648, 2606:2800:220:1:248:1893:25c8:1946
	29, 2601:200:c100:f730:c07c:ff8a:3166:c648, 2606:2800:220:1:248:1893:25c8:1946

ass1_2.pcap:

1, 192.168.80.223, 192.168.80.84
2, 192.168.80.223, 192.168.80.84
3, 192.168.80.223, 192.168.80.84
4, 10.6.102.254, 192.168.80.84
5, 10.6.102.254, 192.168.80.84
6, 10.5.102.254, 192.168.80.84
7, 10.176.253.172, 192.168.80.84
8, 10.176.253.172, 192.168.80.84
9, 10.166.72.77, 192.168.80.84
10, 10.166.72.77, 192.168.80.84
11, 10.166.72.77, 192.168.80.84
12, 10.166.72.82, 192.168.80.84
13, 10.166.72.82, 192.168.80.84
14, 10.166.72.82, 192.168.80.84
15, 10.177.13.84, 192.168.80.84
16, 10.177.13.84, 192.168.80.84
17, 10.177.13.84, 192.168.80.84
18, 10.166.112.1, 192.168.80.84
19, 10.166.112.1, 192.168.80.84
20, 10.166.112.1, 192.168.80.84
21, 152.195.84.214, 192.168.80.84
22, 152.195.84.214, 192.168.80.84
23, 152.195.84.214, 192.168.80.84
24, 152.195.84.139, 192.168.80.84
25, 152.195.84.141, 192.168.80.84
26, 152.195.84.141, 192.168.80.84
27, 93.184.216.34, 192.168.80.84
28, 93.184.216.34, 192.168.80.84
29, 93.184.216.34, 192.168.80.84
30, 93.184.216.34, 192.168.80.84
31, 93.184.216.34, 192.168.80.84
32, 93.184.216.34, 192.168.80.84

ass1_3.pcap:	1, 10.0.0.1, 10.0.0.252
	2, 10.0.0.1, 10.0.0.252
	3, 10.0.0.1, 10.0.0.252
	4, 96.120.14.125, 10.0.0.252
	5, 96.120.14.125, 10.0.0.252
	6, 96.120.14.125, 10.0.0.252
	7, 96.110.222.141, 10.0.0.252
	8, 96.110.222.141, 10.0.0.252
	9, 96.110.222.141, 10.0.0.252
	10, 68.85.120.246, 10.0.0.252
	11, 68.85.120.246, 10.0.0.252
	12, 68.85.120.246, 10.0.0.252
	13, 96.110.45.237, 10.0.0.252
	14, 96.110.45.237, 10.0.0.252
	15, 96.110.45.225, 10.0.0.252
	16, 96.110.33.10, 10.0.0.252
	17, 96.110.33.10, 10.0.0.252
	18, 96.110.33.2, 10.0.0.252
	19, 50.242.151.138, 10.0.0.252
	20, 50.242.151.138, 10.0.0.252
	21, 50.242.151.138, 10.0.0.252
	22, 152.195.77.129, 10.0.0.252
	23, 152.195.77.129, 10.0.0.252
	24, 152.195.77.129, 10.0.0.252
	25, 93.184.216.34, 10.0.0.252
	26, 93.184.216.34, 10.0.0.252
	27, 93.184.216.34, 10.0.0.252
	28, 93.184.216.34, 10.0.0.252
	29, 93.184.216.34, 10.0.0.252
	30, 93.184.216.34, 10.0.0.252

4 References

Chat GPT-3.5 Session Link Pcap 1
Chat GPT-3.5 Session Link Pcap 2 and 3
OpenAI. (2023). ChatGPT [Large language model]. https://chat.openai.com

Request for Comments 792